ARMY RESEARCH LABORATORY

Extending Case-Based Reasoning (CBR) Approaches to Semi-automated Network Alert Reporting

by Robert F. Erbacher and Steve E. Hutchinson

ARL-TR-6410 **April 2013**

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

Army Research Laboratory
Adelphi, MD 20783-1197

ARL-TR-6410 **April 2013**

Extending Case-Based Reasoning (CBR) Approaches to Semi-automated Network Alert Reporting

Robert F. Erbacher
Computational and Information Sciences Directorate, ARL

Steve E. Hutchinson
ICF Jacob and Sundstrom

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
April 2013	Final	June 2011 to August 2011

4. TITLE AND SUBTITLE
Extending Case-Based Reasoning (CBR) Approaches to Semi-automated Network Alert Reporting

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Robert F. Erbacher and Steve E. Hutchinson

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army Research Laboratory
ATTN: RDRL-CIN-S
2800 Powder Mill Road
Adelphi, MD 20783-1197

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TR-6410

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
A substantial amount of cyber security analyst time is spent handling well-known and naïve threats and policy violations on the local network. This includes both the time spent actually identifying and analyzing the activity as well as generating and filing reports associated with the activity. With increasing concern over advanced persistent threats, there is an interest in the development of techniques to automatically handle well-known threats and policy violations. We propose extensions to existing case-based reasoning approaches to support the unique requirements of cybersecurity report generation. Specifically, we consider the fact that we are reporting on hostile actors that will attempt to game the system or manipulate the system to actually aid the actors in obfuscating their activity. In this report, we describe the need for automated reporting, the applicability of case-based reasoning, our proposed extension to the standard case-based reasoning system model, and provide examples of the modified case-based reasoning system as applied to example cybersecurity scenarios.

15. SUBJECT TERMS
Case-based reasoning, semi-automation, report generation, alert correlation

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Robert F. Erbacher
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UU	56	19b. TELEPHONE NUMBER (*Include area code*) (301) 394-1674

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures

INTENTIONALLY LEFT BLANK.

1. Introduction

Computer security typically consists of four primary stages: comprehension, protection, identification, and remediation. Comprehension amounts to the range of research working to identify vulnerabilities, types of attack, sources of attack, etc. Protection is likely the most well-known component since, by protecting the network and its associated systems, we alleviate any potential for compromise. The third component, identification, amounts to the identification that some form of compromise or violation has occurred and must be remediated. In this report, we are concerned with this third component. More specifically, we are concerned with the most common form of identification, namely intrusion detection systems.

The most well-discussed problem associated with intrusion detection systems (IDS) (*3*, *6*) is the need to handle large numbers of false positives and false negatives. A less obvious problem arises from the sheer number of alerts being generated by intrusion detection systems, i.e., true positives. These alerts *must* be remediated by the analyst; they cannot simply be ignored. This is true even for the most unsophisticated of attacks identified by the IDS. This requires that a report be generated for each of these alerts. The process of acquiring and validating the necessary evidence, along with filling in the required report, is time consuming. While critical for many attacks, this usage of time can feel like a waste for naïve and well-known attacks.

The goal with this research is to identify mechanisms by which these naïve and well-known attack alerts can automatically be handled by a Case-Based Reasoning (CBR) system (*1*). Our proposed CBR system provides multiple stages of analysis but ultimately removes the tedious work of handling the majority of unsophisticated attacks. The proposed system will use CBR to match the current alert to the well-known alerts in the knowledge base and use an associated template to fill in the report form. Typically, report generation requires that the analyst enter a wide array of data identifying:

- Affected hosts
- Analyst doing the review
- Attack classification or categorization
- Attack severity
- Evidence of the attack
- Identification of the individual responsible for the affected machine

The incident reporting form for the state of Texas (*29*) provides an example of a more substantial report format. It requires the entry of 34–42 text boxes, depending on the amount of additional information needing to be entered, and three check box groups. While most incident response

forms are not so extensive, the Texas document exemplifies the amount of effort that must be incurred to handle incident reporting. It also exemplifies the information the CBR system must be able to acquire to properly fill in the form. If the form cannot be filled in then it will fall to the analyst to complete the process, again placing time demands on them. A more representative example is shown in figure 1, which shows an example of Request Tracker for Incident Response (RTIR) by Best Practical Solutions LLC (*30, 31*).

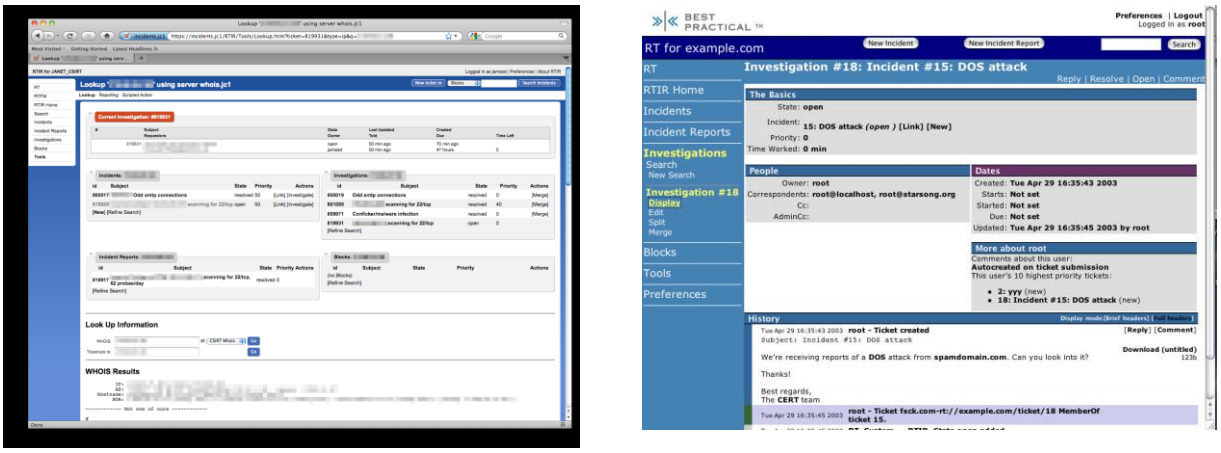

Figure 1. Incident reporting example taken from the RTIR by Best Practical Solutions LLC. The figure shows examples of incident management and the information entered during an investigation for complete reporting.

2. Previous Work

In this research, we explored the ability to automatically generate incident reports using CBR based on a central knowledge base. While no prior work has examined report generation for network incident reports, there is existing research from the data mining field by Vesanto et al. (*27*). The work by Vesanto et al. focuses on generating reports documenting the characteristics of the data after data mining algorithms have executed to provide a starting point for understanding the data and the results of the data mining operations.

Antonides et al. (*2*) looked at streamlining incident reporting for network security incidents. Antonides et al. focused on the issues of non-standardized formats, mediums, and timeframes. While of relevance, this work did not deal with the issue of automating the report generation.

2.1 The Network Alerts Domain

Most IDSs will generate alerts of one form or another; it is actually unusual for an IDS to not generate alerts. The difficulty arises from the fact that all IDSs use different formats for their alerts, leading to correlation difficulties. For general purpose automation of alert generation, these correlation issues will need to be resolved. For our needs, we will be able to focus on a

single format. The most well-known alert format is that of Snort (*18*). Appendices A, E, and G show examples of Snort alerts. Obviously, there are too many varieties of Snort alerts to exemplify here; however, this example shows the details that need to be parsed by an analyst. In essence, these two example sets demonstrate different ranges of capabilities by two different rule sets. Appendix A represents a set of rules provided in the public domain by bleedingsnort.com. The second set of rules, appendix E, is provided in the public domain to registered users by snort.org. While fully explaining the full rule configuration capability of Snort is well beyond the scope of this report, there are several arguments that are absolutely critical to our needs, namely:

- classtype – The classtype argument essentially identifies a broad category or classification in which the alert and associated attack can be placed. This is one step in identifying what attack took place and detailing the contents of the incident report. As is seen from appendices A and E, the classtype is extremely generic and inadequate for complete discrimination.
- sid – The sid is a unique identifier associated with each rule. The sid can be extremely valuable in associating additional information with an alert.
- msg – The msg is additional message information that should be displayed in association with the fired alert. The msg essentially allows the rule to provide the analyst with an immediately interpretable English representation of the alert. (Snort-rule message examples are shown in appendix H.)
- reference – The reference can be viewed as validation or evidence as to the meaning of the identified event sequence, and why that event sequence should be considered an attack or policy violation.

Of note is the fact that most alert generating tools, like with Snort, will provide some method of specifying similar types of information—though often not as distinctly. For instance, the Bro Network Security Monitor (*0*) provides very limited support but can be adapted through scripting to provide equivalent support. In essence, Bro provides a single message when an alert is raised, namely: ″event <string>″, though more extensive capabilities are provided to create alert rules. This string could essentially embed the same information as Snort.

The bleedingsnort.com associated alerts provide the epitome of our goal with this work. This is due to the fact that the unique sid numbers intrinsic to the Snort rules (appendix A) are associated with extensive documentation (appendices B–D). Appendix B provides an example of a documentation example with the least information and the most generic description. Appendix D provides the most information and the most specific explanation. Additionally, appendix D provides an external reference for further validation, a necessary requirement when enforcing remediation of a system.

While bleedingsnort.com provides extremely detailed descriptions, the rules (appendix E) from snort.org for registered users provide far less information. There is still additional information

associated with the unique sid numbers (appendix F). However, this documentation is completely inadequate. In addition, most Snort installations would likely not even have any such additional information available. This is exemplified by Snort preprocessor-generated alerts, which will typically not have a sid number or even a classification label; identification of which preprocessor rule fired is the extent of classification provided. With general rules, even if the classification and sid are not included, the message associated with the alert provides some level of discernment, the hope being that these message are actually unique and descriptive. This cannot be guaranteed with the thousands of rules in a typical IDS configuration.

This deviation in documentation exemplifies the goals of this research. Intrinsically, the goal is to reduce the workload on the analyst when dealing with naïve and well-known attack alerts. Fundamentally, this means filling in the report forms, identified in the introduction, automatically. However, a secondary goal will be to avoid requiring analysts to create and maintain the extensive set of documentation demonstrated by bleedingsnort.com. The desired solution will support all scenarios from the extremely detailed information being provided as is demonstrated by bleeding Snort, to the scenario in which an alert generation tool, such as Bro, is used and very little additional information has been provided in the alert message or associated documentation. Initially, a site will have very few templates in the knowledge-base, but as events are identified the knowledge-base will grow; clearly, the most common events will get templates first showing a very rapid decline in analyst time expenditure on naïve events.

It is for this reason that we examine CBR. Such CBR systems are intrinsically designed to handle this variation in documentation automatically and will avoid the need for analysts to independently maintain a documented database of information. This greatly resolves the need for analysts to waste time on naïve and well-known attacks.

2.2 Case-Based Reasoning (CBR)

CBR originally derived from a cognitive science-based model of dynamic memory by Schank et al. (*19*). Schank's work resulted in the first two computer-based CBR systems in 1983 (*11, 13*). The fundamental process model, figure 2, was identified by Aamodt et al. (*1*) in 1994. It is this process model, termed the four REs, that has since driven the majority of research and application development in CBR research. Fundamentally, the model is based on the incorporation of a knowledge base (*22*), or case base, which is a historical set of scenarios for which solutions are known. A new scenario is then matched against the existing scenarios to find the most relevant match, which is then mapped to the new scenario providing an updated solution. Such CBR has been applied to a wide range of domains including:

- Breathalyzers (*7*)
- Bronchiolitis (*7*)
- E-Clinic (*7*)

- Intelligent tutoring systems (*21*)
- Help desk systems, i.e. diagnosis (*22*)
- Electronic commerce product recommendation systems (*22*)
- Classification, i.e., class membership (*22*)

An example provided in (*32*) exemplified this through the application of CBR to automobile repair, figure 3.

Known Case	Known Case		New Case
Description: Problem: Headlight not working Car: {name} Year: {year} State of lights: OK State of light switch: OK **Solution:** Diagnosis: Headlight fuse defect Repair: Replace headlight fuse	**Description:** Problem: Headlight not working Car: {name} Year: {year} State of lights: surface damaged State of light switch: OK **Solution:** Diagnosis: Bulb defect Repair: Replace front light		**Description:** Problem: Brake light not working Car: {name} Year: {year} State of brake lights: OK State of light switch: OK **Solution:** Diagnosis: Brake light fuse defect Repair: replace break light fuse

Figure 2. Example showing the application of a known case to a new case with adaptation of an existing template (*32*).

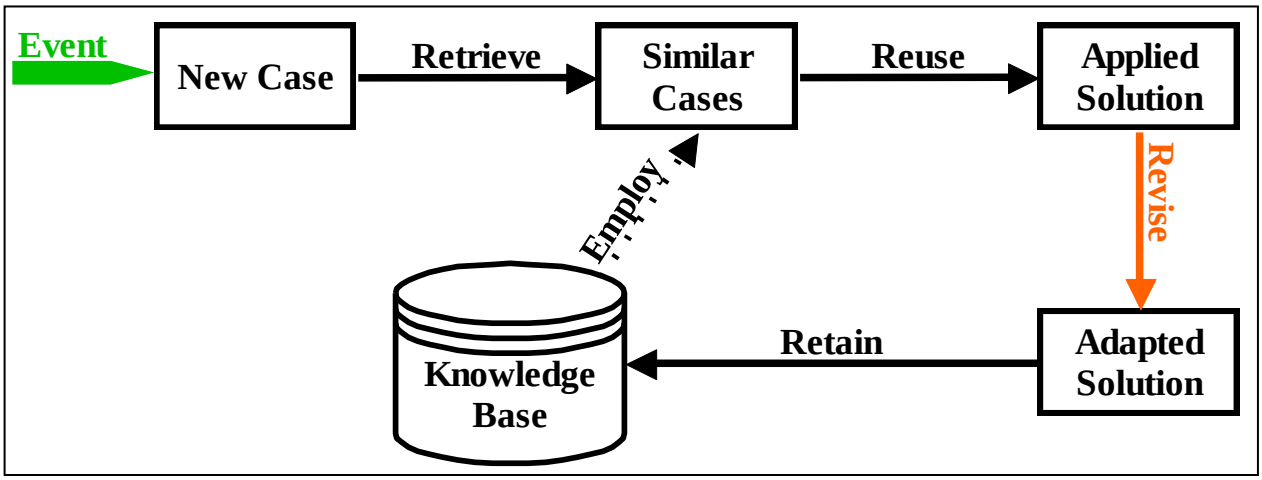

Figure 3. Traditional CBR process model.

There have been many surveys and reviews of the existing literature related to CBR (*0*), an indication of the extent of research performed in this domain. Each of the four main tasks in CBR has an extensive research following, though the research in retrieval is likely the most extensive due to the critical importance in the returned results.

2.3 Similarity Metrics and Retrieval Criteria

Similarity metrics are the most basic form of retrieval mechanism; however, these similarity metrics are typically integrated with additional metrics to improve the usefulness of the returned results. A survey/taxonomy of similarity metrics can be found in Cunningham (*5*). Examples of research in retrieval mechanisms include:

- Information theory approaches (*17*). This research provides for the identification of similarities between instances in and ontology.
- User defined functions (*24*). The associated patent also covers the representative database issues.
- Abduction versus deduction (*25*).
- Fuzzy similarity (*25*).
- Contextual probability (*28*). This metric integrates probability with distance-based neighborhood weighting, and works for both ordinal and nominal data.
- Adaptive similarity (*14*). This paradigm allows for specification of new similarity metrics and identification of the similarity metric to be applied in particular scenarios without the need for reprogramming.
- Semantic vs. syntactical similarities (*1*).
- Models of similarity (*16*). The goal of this work was to identify the primary classes of similarity, including absolute, relative, and metric.

Specific similarity metrics for categorical data include (*4*):

- Overlap
- Eskin
- IOF
- OF
- Lin
- Lin1
- Goodall1
- Goodall2
- Goodall3
- Goodall4

- Smirnov
- Gambaryan
- Burnaby
- Anderberg
- Neighborhood Counting Metric (*28*)

3. Challenges of the Network Alerts Domain

The network alerts domain provides several fundamental challenges that had, at least in part, to be dealt with during the course of this research. While we focus on Snort alerts in the majority of this report and use its parameters for the similarity metrics, we must consider that the goal is to support any IDS and, therefore, must be prepared to handle any type of parameter. Future research should examine more complete resolutions to these problems. The first challenge is the identification of parameters in network alerts, themselves. This becomes problematic due to the categorized nature of nearly all attributes associated with network alerts. For instance:

- Date – An ordinal type.
- Time – An ordinal type.
- Alert type – A categorical type. The numerical types and subtypes will often be incomprehensible, especially given the number of types of alerts. It is the textual representation that can be immediately discerned for what it is. The types + subtypes do create a hierarchical type, but in this case the hierarchy typically is of no concern. It is the specific alert type that is interpreted for prioritization and remediation strategies. The hierarchical information can be helpful in terms of identification of similarity groups if subtypes do not match exactly.
- IP addresses – A hierarchical type. IP addresses outside of the same domain have no relationship. The extent of similarity essentially amounts to how many of the decimals in order from left to right are identical. The actual numerical values within each grouping are categorical, having no actual relationship to one another; e.g.,
    - 62.0.0.0 – Domain located in Israel.
    - 63.0.0.0 – Domain located in Virginia.
    - These two addresses can only be considered similar if we are considering that they are both on Earth.

- Port numbers – Port numbers are particularly problematic since they are categorical but potentially have 65,000 different values. These values are categorical since the numerical associations have no actual meaning other than to assign a meaning in the digital computer domain. The question then becomes do we treat these values independently or lump them into groups? The solution with port numbers is to either not use them or look for an exact match. This does maintain the problem of what to do if port randomization of some form is used. This is where the rest of the packet comes into play; if the payload is indicative of SSH but the port number is not, then we have an anomaly worthy of an alert. However, as far as this work is concerned, we are not attempting to perform anomaly detection or interpretation of attacker behavior/intent. Therefore, we must solely rely on the individual components—i.e., payload indicative of an ssh attack *or* port indicative of an ssh attack.

Osborne et al. (*16*) discuss models of similarity for use with CBR that covers organizations of similarity metrics for the different attribute types. This results in a challenge as far as similarity metrics, since we will need multiple metrics to handle categorical, ordinal, and hierarchical data types. Ultimately, the resultant similarity value for an alert will be a weighted combination of the similarity metrics of each individual attribute of the alert.

4. Application of CBR to Network Alerts

With the goal of reducing analyst effort required to handle naïve and well-known attacks, we examine the applicability of CBR-based systems to the task of correlating alerts and automating report generation. In the simplest case, we can use the sid numbers associated with the Snort alert to acquire the detailed description from a database. However, such a scenario will not be effective in the general case, in which such information is not available. Relying on sid numbers will also not support the case of slight variations in an attack that may cause the rule to fire, but the existing data is inadequate in a fixed form. Finally, since the goal is to reduce analyst effort, requiring analysts to manually maintain such databases of attacks and associated report templates. As discussed in the previous work session, these are tasks that CBR-based systems are specifically designed for. The unique nature of network analyst work requires adaptations of the typical 4R model associated with CBR-based systems (figure 4).

- **Retrieve** – The retrieve function is much more complex than a traditional CBR-based retrieve function due to the fact that we may need to retrieve based on multiple alerts (correlation) and the complexity of the information being searched. The amount of detail we have from the alert will aid in determining what similarity metric should be employed; if we can use a simple overlay metric then we do not want to use a more complex error-prone metric.

- **Reuse** – The application of the retrieved template from the database will not change from the traditional function.
- **Revise** – Since alerts did fire on the specified event, we know that the attack has been seen before and rules generated. However, this could be a slight variation of known malicious activity and will most likely involve different IP addresses. The revise function should be able to automatically fill in the template with both the static information from the template and the dynamic information from the current alerts. This will then need to be revised further by the analyst for accuracy, additional validation for the new variant, etc.
- At this point, the analyst must determine whether to send the adapted solution (generated report) or not.
- **Retain** – If the new case has important modifications over existing cases—i.e., it's a new variant of a known malicious attack—then the case should be learned by the system and injected into the knowledge base to simplify report generation of the next occurrence of this variant. However, this must be approved by the analyst. This is due to the potential for an attacker to game the system if it were fully automated. The analyst must determine if injecting the new variant into the knowledge base will impact future interpretation of events by missing critical activity; i.e., not seeing a particular alert could impact an analyst's world view and impact detection of persistent sophisticated threats.
- Whether the new event is integrated into the knowledge base or not, the knowledge of the new attack characteristics will be added to the analyst's domain expertise that can be further employed by that analyst in future alert analysis scenarios—for instance, identifying trends in how variants of known malicious events are being generated. This type of trending analysis cannot currently be integrated into CBR-based systems and is the reason the analyst must remain firmly integrated into the analysis and report generation process, even if the tedious aspects of this process are handled automatically.

Existing research has examined the issue of removing harmful cases from the case base, such as duplicate entries. However, the need for the analyst to manually identify whether a case should be added to the knowledge base is unique. These cases may not be harmful per se, but may simply be alerts that the analyst must be aware of and handle directly. There may be cases, for instance, that can too easily obfuscate non-naïve attacks that should not be handled automatically. Similarly, we have determined that we do not wish to exhaust analyst time by forcing them to perform even more labor by validating the proposed report. Doyle (*7*) proposed a mechanism by which explanatory text can be generated within CBR systems. Similarly, Silva et al. (*20*) discuss the integration of arguments into the knowledge base.

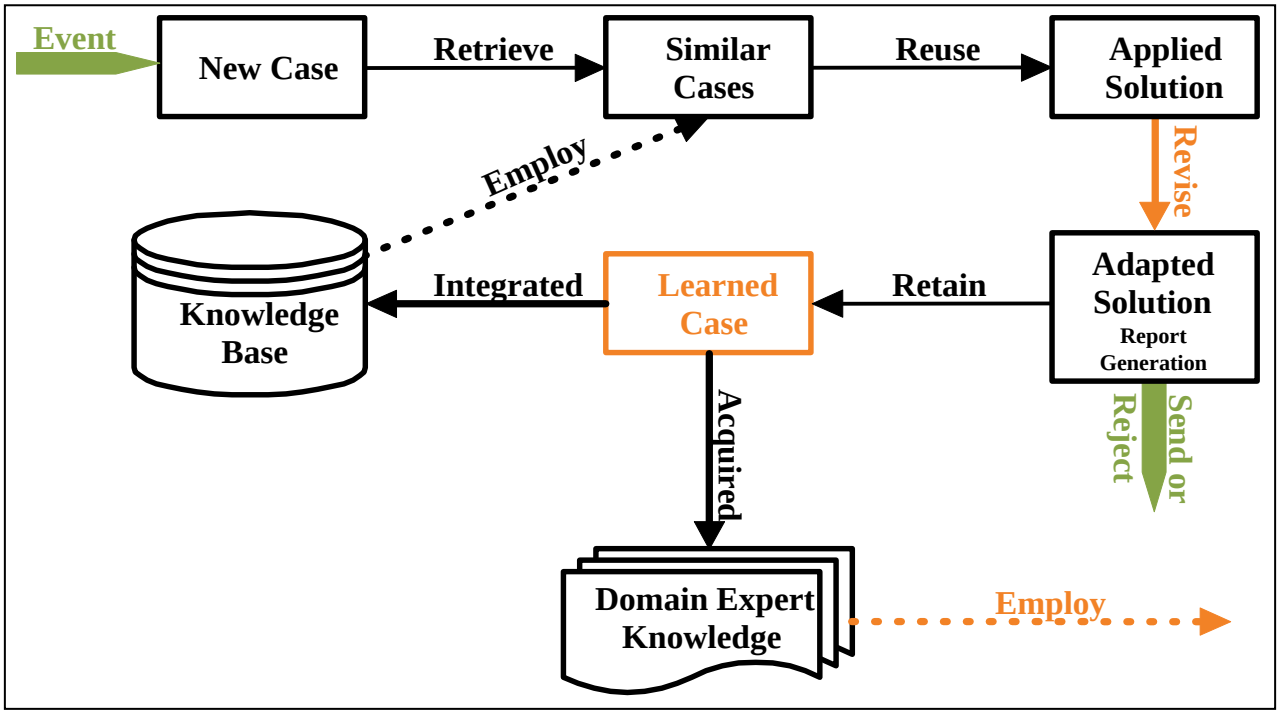

Figure 4. CBR process model extended to the unique characteristics of network security incident report generation. System IO is emphasized as green. Analyst interaction and decision making is emphasized in orange.

5. CBR Process Applied to Network Alert Analysis

The proposed process flow diagram associated with our new model is presented in figure 5. The critical components include the following:

- The use of the knowledge-base to identify if the received event is typically part of a sequence of events. This is necessary to identify sophisticated persistent events of which we are only seeing a single step. This is then passed onto the alert correlation tool to locate and integrate any other events. Since we already know, from the knowledge-base, what events we are looking for this process should make the alert correlation task simpler.
- If we get results that differ from the expected, we hand the process off to an analyst. So if we got a single event but the event is usually part of the sequence, we must alert the analyst so they can ensure the event is handled correctly, whether it is actually just a solo event or not. Events are also passed on to the analyst if there is no usable template. This occurs if there is no similar alert at all or the performance metrics of the similarity metrics are not within threshold.
- The application of the similarity metric is actually an iterative process in which different parameters are employed until a statistically relevant result is found, or all parameters are exhausted.

- The analyst is asked for verification before the generated report is sent or integrated into the knowledge-base. This is to prevent gaming of the system by an attacker. The majority of the effort is still handled automatically as the report is filled in by the CBR-based system.
- The remaining components of the process essentially revolve around the CBR-based model.

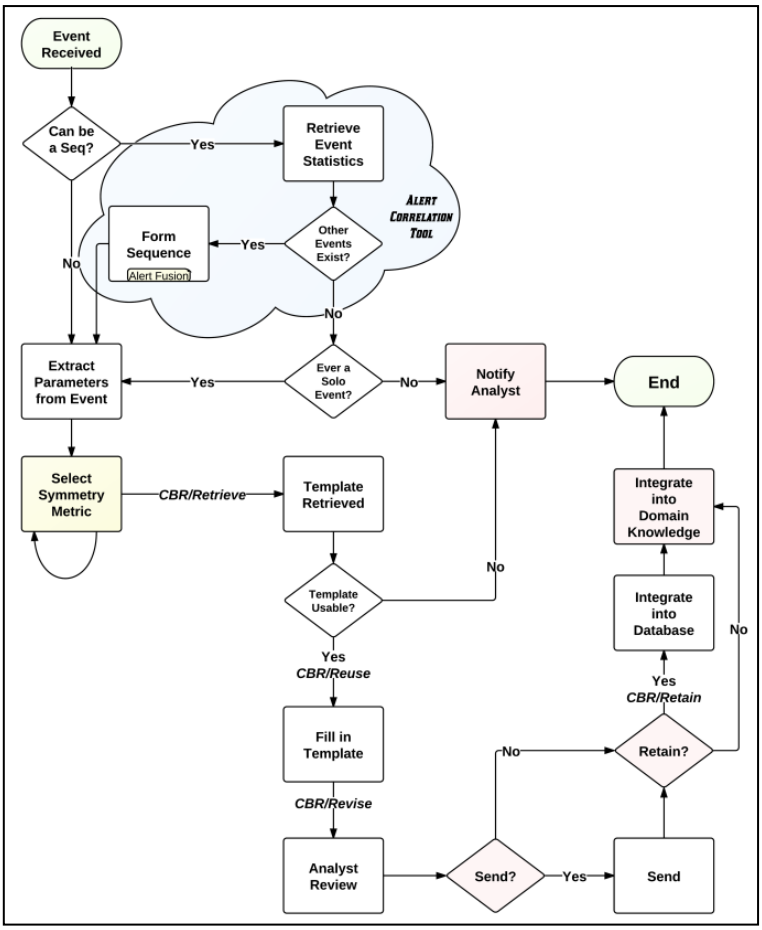

Figure 5. Process flow diagram showing the detailed steps in our modified CBR process. Analyst involvement is in the light red steps. Green steps are the start and end conditions. Yellow identifies sub-processes; i.e., the similarity metric selection is exemplified in figure 6. The light blue is where interfacing with the alert correlation tool would occur.

The similarity metric process is examined in detail in figure 6. The key characteristics are the repetitive process of selecting parameters by pre-determined weighting, applying the most relevant similarity metric, and extracting the relevant data subsets based on that metric. Depending on the specific metric and scenario, there may be multiple results returned from a single pass. Since we need to reduce our selection to a single entry, we must perform multiple passes to extract a single result. In some cases it is, of course, possible to come up with more

than one exact match after exhausting all of the parameters. This may be an indication that templates were added to the knowledge-base that did not need to be, or we are dealing with a very generic alert. The basic solution is to grab the first match, the most recently updated match, etc.

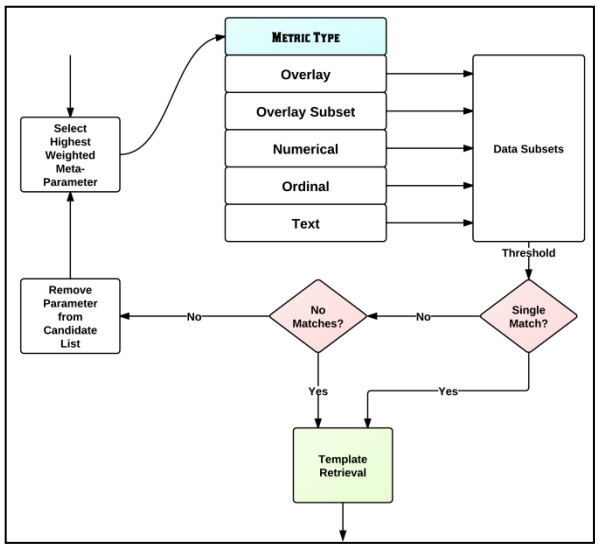

Figure 6. Exemplifying the similarity metric process. The basic process is iterative as the different metrics are applied based on the parameter type. Parameters are selected based on weighting.

6. Validation

Given we currently only have a theoretical model, we performed case studies to validate the model. The validation based on the model will help to ensure the likelihood of success from a full implementation.

Scenario 1: Alert exists in knowledge-base

Steps:

1. Alert generated with message M_1="BLEEDING-EDGE MALWARE 180solutions Update Engine"

   (*Begin Retrieve*)

2. Perform search using overlay metric for message M_1.

   a. Binary similarity metric

   b. Only return exact matches

c. Returns a single exact match

3. Calculate F-measure values to identify confidence to analyst

   a. tp (true positives) = 1

   b. fp (false positives) = 0

   c. fn (false negatives) = 0

   d. tn (true negatives) = 3058

   e. precision [0] = $\frac{tp}{tp + fp}$ = 1

   f. recall [0] = $\frac{tp}{tp + fn}$ = 1

   g. accuracy [0] = $\frac{tp + tn}{tp + fp + fn + tn}$ = 1

   h. F1 (F-measure) [0] = $2 \cdot \frac{precision \cdot recall}{precision + recall}$ = 1

   i. Conciseness [0][0][0] = $\frac{tp + fp}{tp + fp + fn + tn} \times 100\%$ = 0.0327%

      The goal with conciseness is to provide an indication as to what percent of the total documents were returned in the results—i.e., how selective or unique are the returned documents. While conciseness is used in the literature, we find selectivity a more representative term. False positives are included in the formula to provide an indication of the true selectivity of the results returned to the user, since the false positives will be additional results the user needs to parse to find the best result.

4. A single match is returned.

   (*End Retrieve*)

   (*Begin Reuse*)

5. Extract report template from knowledge base.

6. Fill in template report with details from this event.

   a. IP Addresses

   b. Data/Time

   c. Analyst performing review

   (*End Reuse*)

   (*Begin Revise*)

7. Analyst examines form for validity, acceptance, applicability, etc.
    a. Does the particular policy affect the designated location?
    b. Is the alert still a reportable incident?
    c. Identify the level of the incident to determine extent of notification required.
8. Modify the report and disseminate or reject the report.

    (*End Revise*)

    (*Begin Retain*)
9. Analyst determines if this event should be specifically stored in knowledge base
10. Store in knowledge-base.

    (*End Retain*)
11. Analyst's domain expertise is enhanced.
12. Analyst's enhanced domain expertise is employed in future incidents.

Scenario 2: Alert does not exist in knowledge-base for one level.

1. Alert generated with message M_1="BLEEDING-EDGE MALWARE 180solutions Spyware (action url reported)"

    (*Begin Retrieve*)
2. Perform search using overlay metric for message M_1
    a. Binary similarity metric
    b. Only return exact matches
    c. No match found. This indicates that while the rule exists, no report template has specifically been integrated into the knowledge-base related to this alert. We must locate a related template to use as a starting point.
3. Remove one word of specification and repeat overlay metric
    a. Assume each word further to the right increases specificity
    b. Assume a parenthetical set segregates one "word"
    c. M2="BLEEDING-EDGE MALWARE 180solutions Spyware"
    d. Matches:

i. P1=BLEEDING-EDGE MALWARE 180solutions Spyware (tracked event reported)

ii. P2=BLEEDING-EDGE MALWARE 180solutions Spyware Reporting

iii. P3=BLEEDING-EDGE MALWARE 180solutions Spyware Keywords Download

iv. P4=BLEEDING-EDGE MALWARE 180solutions Spyware Install

v. P5=BLEEDING-EDGE MALWARE 180solutions Spyware Defs Download

vi. P6=BLEEDING-EDGE MALWARE 180solutions Spyware config Download

vii. P7=BLEEDING-EDGE MALWARE 180solutions Spyware versionconfig POST

4. Apply overlay metric to classtype for sanity check validation and generate reduced subset
   a. All potential matches P_i match classtype
5. Traverse Snort alert rule argument list
   a. Accumulate similarity metrics for each potential match
      i. +1 for exact match
      ii. –1 for arguments that do not exist or extra arguments in potential match
      iii. +SMV. Otherwise add the value from the similarity metric
      iv. Return the potential matches in the order from highest score to lowest
   b. Ex=the xth argument for event E.
   c. E1,1=tcp; E1,2=$HOME_NET; E1,3=any; E1,4=$EXTERNAL_NET; E1,5=$HTTP_PORTS; E1,6=flow: to_server,established; E1,7=uricontent:"/actionurls/ActionUrl"; E1,8=nocase; E1,9=reference:url, securityresponse.symantec.com/avcenter/venc/data/pf/adware.180search.html;
   d. Ax,y=argument. Where x is the potential match number and y is the argument number.
   e. We must search for the existence of each argument; naively, this becomes n×m in the number of arguments in the event and potential match. Remove duplicate and grammatical arguments. This loses context but becomes much simpler. While many of the parameters shown later seem fairly generic, keep in mind the goal is to demonstrate a completely generic and broadly applicable approach.
   f. A1,1=alert; A1,2=tcp; A1,3=$HOME_NET; A1,4=any; A1,5=->; A1,6=$EXTERNAL_NET; A1,7=$HTTP_PORTS; A1,8=msg: BLEEDING-EDGE MALWARE 180solutions Spyware (tracked event reported) A1,9=flow: to_server,established; A1,10=uricontent:"TrackedEvent.aspx?"; A1,11=nocase;

A1,12=uricontent:”eid=”; A1,13=nocase; A1,14=reference:url, securityresponse.symantec.com/avcenter/venc/data/pf/adware.180search.html; A1,15=classtype: trojan-activity; A1,16=sid: 2001397; A1,17=rev:6;

i. Remove A1,1; A1,5; A1,8; A1,13; A1,15; A1,16; A1,17; as being duplicates, already checked, and non-discriminable.

ii. Note that each of these arguments is essentially textual, i.e., ordinal in nature.

iii. An overlay-based textual similarity metric will be used.

g. A1,2=tcp; A1,3=$HOME_NET; A1,4=any; A1,6=$EXTERNAL_NET; A1,7=$HTTP_PORTS; A1,9=flow: to_server,established; A1,10=uricontext:”TrackedEvent.aspx?”; A1,11=nocase; A1,12=uricontent:”eid=”; A1,14=reference:url, securityresponse.symantec.com/avcenter/venc/data/pf/adware.180search.html;

i. Renumber

h. A1,1=tcp; A1,2=$HOME_NET; A1,3=any; A1,4=$EXTERNAL_NET; A1,5=$HTTP_PORTS; A1,6=flow: to_server,established; A1,7=uricontext:”TrackedEvent.aspx?”; A1,8=nocase; A1,9=uricontent:”eid=”; A1,10=reference:url, securityresponse.symantec.com/avcenter/venc/data/pf/adware.180search.html;

i. A2,1=tcp; A2,2=$HOME_NET; A2,3=any; A2,4=$EXTERNAL_NET; A2,5=$HTTP_PORTS; A2,6=flow: to_server,established; A2,7=uricontext:”/showme.aspx?keyword=”; A2,8=nocase; A2,9=reference:url, securityresponse.symantec.com/avcenter/venc/data/pf/adware.180search.html;

j. A3,1=tcp; A2,2=$HOME_NET; A3,3=any; A3,4=$EXTERNAL_NET; A3,5=$HTTP_PORTS; A3,6=flow: to_server,established; A3,7=uricontext:”keywords/kyf”; A3,8=nocase; A3,9=content:”partner_id=”; A3,10=reference:url, securityresponse.symantec.com/avcenter/venc/data/pf/adware.180search.html;

k. A4,1=tcp; A4,2=$HOME_NET; A4,3=any; A4,4=$EXTERNAL_NET; A4,5=$HTTP_PORTS; A4,6=flow: to_server,established; A4,7=uricontext:”/downloads/installers”; A4,8=nocase; A4,9=content:”simpleinternet/180sainstaller.exe”; A4,10=reference:url, securityresponse.symantec.com/avcenter/venc/data/pf/adware.180search.html;

l. A5,1=tcp; A5,2=$HOME_NET; A5,3=any; A5,4=$EXTERNAL_NET; A5,5=$HTTP_PORTS; A5,6=flow: to_server,established; A5,7=uricontext:”/geodefs/gdf”;

A5,8=nocase; A5,9=reference:url,
securityresponse.symantec.com/avcenter/venc/data/pf/adware.180search.html;

m. A6,1=tcp; A6,2=$HOME_NET; A6,3=any; A6,4=$EXTERNAL_NET;
A6,5=$HTTP_PORTS; A6,6=flow: to_server,established;
A6,7=uricontext:"/config.aspx?did="; A6,8=nocase; A6,9=reference:url,
securityresponse.symantec.com/avcenter/venc/data/pf/adware.180search.html;

n. A7,1=tcp; A7,2=$HOME_NET; A7,3=any; A7,4=$EXTERNAL_NET;
A7,5=$HTTP_PORTS; A7,6=flow: to_server,established;
A7,7=uricontext:"/versionconfig.aspx?"; A7,8=uricontext:"&ver="; A7,9=nocase;
A7,10=reference:url,
securityresponse.symantec.com/avcenter/venc/data/pf/adware.180search.html;

o. For brevity, consider the following table of results in terms of the similarity metric results for each of the arguments.

	$A_{x,1}$	$A_{x,2}$	$A_{x,3}$	$A_{x,4}$	$A_{x,5}$	$A_{x,6}$	$A_{x,7}$	$A_{x,8}$	$A_{x,9}$	$A_{x,10}$	**SUM**	**Rev**	**Priority Order**
P_1	1	1	1	1	1	1	0	1	-1	1	7	6	4
P_2	1	1	1	1	1	1	0	1	1	0	8	5	1
P_3	1	1	1	1	1	1	0	1	-1	1	7	3	6
P_4	1	1	1	1	1	1	0	1	-1	1	7	4	5
P_5	1	1	1	1	1	1	0	1	1	0	8	3	2
P_6	1	1	1	1	1	1	0	1	1	0	8	2	3
P_7	1	1	1	1	1	1	0	-1	1	1	7	1	7

p. Given the number of duplicate scores, we can use two additional metrics for ordering. First, the revision number associated with the Snort rule will provide an indication of the currency and accuracy of the rule. Second, the old standby of using the ordering of the rules can be used.

6. A list of matches is returned in priority order.

   a. Cycle through the matches in priority order.

   b. Identify the highest priority match that has an associated template in the knowledge base.

   c. Return this highest priority match.

   d. In the case that no matches have a template in the knowledge-base, return to step 3 and repeat.

Efficiency note: in an efficient implementation the matches without a template would obviously be pre-filtered.

7. Calculate F-measure values to identify confidence to the analyst. There were three documents that had equivalent similarity metrics of 8. The differentiation between these, using order and revision, is essentially artificial and a direct indication of uncertainty; i.e., we could just have easily chosen randomly. The four documents with similarity metrics of 7 can be construed as false positives. Thus,

    a. tp = 3

    b. fp = 4

    c. fn = 0

    d. tn = 3052

    e. precision = 0.4286

    f. recall = 1

    g. accuracy = 0.9987

    h. F1 = 0.6

    i. conciseness = 0.2288%

8. A single match is returned.

    (*End Retrieve*)

    (*Begin Reuse*)

9. Extract the report template from knowledge base.

10. Fill in the template report with details from this event.

    a. IP Addresses

    b. Data/Time

    c. Analyst performing review

    (End Reuse)

    (Begin Revise)

11. Analyst examines form for validity, acceptance, applicability, etc.

    a. Does the particular policy affect the designated location?

    b. Is the alert still a reportable incident?

    c. Identify the level of the incident to determine extent of notification required.

12. Modify the report and disseminate or reject the report.

    (*End Revise*)

    (*Begin Retain*)

13. Analyst determines if this event should be specifically stored in the knowledge-base.
14. Store in the knowledge-base.

    (*End Retain*)

15. Analyst's domain expertise is enhanced.
16. Analyst's enhanced domain expertise is employed in future incidents.

For comparison, consider the results of the conciseness metric for two further iterations of the algorithm:

	Metric			Iteration 3	Iteration 4
a.	tp	=	9		3059
b.	fp	=	N/A		N/A
c.	fn	=	0		0
d.	tn	=	3050		0
e.	conciseness	=		0.2942%	100.00%

This exemplifies the impact and value of the conciseness metric. While extreme in this case, the issue is to provide a representation to the analysts of when the returned results may not be of value. Clearly, there will be jumps in the conciseness metric that clearly indicate a lack of selectivity that makes the returned results ineffective even when such jumps are not as extreme as this scenario; the need is to provide the fact that such a jump did, in fact, occur to the analyst. Since the total number of potential results may vary, the percentage associated with the conciseness value is more informative. For instance, the percentage will be informative whether there are 20 total potential results or 100,000.

Scenario 3: Why Semi-Automated, Denial of Reporting

```
alert udp any 53 -> any any (msg: "BLEEDING-EDGE Suspicious DNS
server answer\: 217.16.26.148"; content:"|d9 10 1a 94|";
classtype: bad-unknown; sid: 2001838; rev:2; )
```

```
alert udp any 53 -> any any (msg: "BLEEDING-EDGE Suspicious DNS
server answer\: 218.38.13.108"; content:"|da 26 0d 6c|";
classtype: bad-unknown; sid: 2001837; rev:3; )
```

Consider the above alerts.

1. Alert generated with message M_1= “BLEEDING-EDGE Suspicious DNS server answer\: 217.16.26.148”
2. Perform search using overlay metric for message M_1.
    a. No acceptable matches
    b. Conciseness too high
3. Perform repeated searches using range of similarity metric for message M_1.
    a. No acceptable matches
    b. F-Measure too low
4. Analyst creates report and injects into the knowledge-base.
5. Alert generated with message M_1= “BLEEDING-EDGE Suspicious DNS server answer\: 217.16.26.148”
6. Perform search using overlay metric for message M_1.
    a. Exact match found
    b. Very low conciseness, 1/3059*100%=0.0327%
    c. Very high F-Measure, F1=1
    d. Metrics within threshold
7. Extract template from the knowledge-base
8. Fill in the template with details on this specific case.
9. Distribute report
10. Delay 25 ms
11. Repeat from step 5
12. Alert generated with message M_1= “BLEEDING-EDGE Suspicious DNS server answer\: 218.38.13.108,”
13. Perform search using overlay metric for message M_1.
    a. No acceptable matches
    b. Scenario 1, matches existing case for “BLEEDING-EDGE Suspicious DNS server answer\: 217.16.26.148” after removing one word

c. Scenario 2, Conciseness too high (above threshold)

14. Scenario 2: Perform repeated searches using range of similarity metric for message M_1
    a. All parameters are exact matches except content
    b. Content matches string comparison, length + similar letters + limited distance between values
15. Extract template from knowledge base
16. Fill in template with details on this specific case
17. Distribute report
18. Delay 25 ms
19. Alert generated with message M_1= "BLEEDING-EDGE Suspicious DNS server answer\: 218.38.13.108"
20. Perform search using overlay metric for message M_1
    a. Exact match found
    b. Very low conciseness, 1/3059*100%=0.0327%
    c. Very high F-Measure, F1=1
    d. Metrics within threshold
21. Extract template from knowledge base
22. Fill in template with details on this specific case
23. Distribute report
24. Delay 25 ms
25. Repeat from step 19

The issue with automating these alerts is that a large number of reports can be generated very quickly. These reports can easily obfuscate real important reports. Sophisticated attackers will already use naïve port scans and naïve attacks to camouflage their attack. The next step is for attackers to use naïve attacks to evade detection after a successful compromise.

Scenario 4: Why Semi-Automated, Event Priority Obfuscation

```
alert tcp $HOME_NET any -> $EXTERNAL_NET !6661:6668 (msg:
"BLEEDING-EDGE ATTACK RESPONSE IRC - DCC file transfer request
on non-std port"; flow: to_server,established; content:"PRIVMSG
"; nocase; offset: 0; depth: 8; content:" \:.DCC SEND"; nocase;
```

```
tag: session,300,seconds; classtype: policy-violation; sid:
2000349; rev:5; )

alert tcp $HOME_NET any -> $EXTERNAL_NET !6661:6668 (msg:
"BLEEDING-EDGE ATTACK RESPONSE IRC - Private message on non-std
port"; flow: to_server,established; dsize: <128;
content:"PRIVMSG "; nocase; offset: 0; depth: 8; tag:
session,300,seconds; classtype: trojan-activity; sid: 2000347;
rev:5; )
```

Note the limited differences in the previous two alert rules, ignoring differences unrelated to the rule process. The first alert has `content:" \:.DCC SEND";` while the second has `dsize: <128;` the result of the rules are the same. Now consider the difference in class type—a `policy-violation` versus `trojan-activity`. The difference in interpretation by the analyst will be enormous. The trojan activity will get very high priority, while the policy violation will get very low priority. In general, it is the after effects of a successful compromise that an analyst can identify and not the attack, itself. Thus, the attack response of the installed trojan is how the presence of the trojan will be identified.

1. Alert generated with message M1= “BLEEDING-EDGE ATTACK RESPONSE IRC - DCC file transfer request on non-std port,”
    a. First time this alert is seen
2. Perform search using overlay metric for message M1.
    a. No acceptable matches
    b. Conciseness too high
3. Perform repeated searches using range of similarity metric for message M1.
    a. No acceptable matches
    b. F-Measure too low
4. Analyst creates the report and injects it into the knowledge-base.
5. New, zero-day, trojan injected into network
    a. Modification of existing Trojan that would normally be identified by rule with sid 2000347
    b. Challenge-response packets modified to contain " \:.DCC SEND" as the first sequence of bytes
6. Alert generated with message M1= “BLEEDING-EDGE ATTACK RESPONSE IRC - DCC file transfer request on non-std port,”

7. Perform search using overlay metric for message M1.
    a. Exact match found on first pass
8. Report automatically generated and distributed with classtype: policy-violation and associated priority.

This rather simple example demonstrated the impact of a fully automated system; note that these are actual rules in the bleeding Snort rule base. Given the tens of thousands or perhaps hundreds of thousands of events in a typical rule-set, it is likely quite common for such similar rules to exist. An analyst reviewing the report before letting it be sent out will identify potential inconsistencies and will raise the priority when multiple alerts start to be generated. The automated system will not notice anything unusual. This example illustrates the ease with which an attacker can de-escalate the priority of their compromise and evade detection. Given the extent to which attackers go through to compromise hosts, we have to assume they will modify compromises in a similar fashion to maintain control of the host for as long as possible. Keep in mind the bleeding Snort rule base is public domain and the attackers will have access to it. While internal rules may differ, attackers will also put a lot more time into evasion than we did, especially nation-states.

7. Conclusions

This research developed a model for the automated generation of reports based on a knowledge base created by analysts over time. While the creation of the reports is automated, we do incorporate the analyst in the process to validate the report before disseminating the report or incorporating the results into the knowledge base. This will prevent attackers from gaming the system. As attackers will analyze the code of an application to identify potential vulnerabilties i.e., buffer overflows, they will similarly attempt to identify vulnerabilities in the analyst process.

The resultant model was validated with case studies. These case studies provide sufficient validation as to the viability of the model to warrant full implementation.

Of note is the reliance on open source rule-systems for Snort, i.e., Bleeding Snort. As we rely solely on open source capabilities, we are limiting the potential sources of attack; i.e., we only assumed Snort alerts were available. In a real environment, multiple tools will likely be available all generating alerts. The similarity metric process, exemplified in figure 6, assumes multiple alert types. This is why we cannot always just assume the overlay metric will be sufficient. We must support all alert types in the automated report generation and correlation capabilities. The current model supports this. The limitation of such generic support is that it provides many more avenues for an attacker to compromise or game the system.

8. References

1. Aamont, Agnar; Plaza, Enric Plaza. Case-based Reasoning: Foundational, Methodological Variations, and System Approaches. *AI Commun.* **7 March 1994**, *1*, 39–59.
2. Antonides, J. R.; Benjamin, D. N.; Feldpausch, D. P.; Salem, J. S. Streamlining the U.S. Army Network Incident Reporting System. *Systems and Information Engineering Design Symposium, 2008. SIEDS 2008. IEEE* 22–25 April 2008, 17–21.
3. Anderson, J. P. *Computer Security Threat Monitoring and Surveillance*. Technical report, James P. Anderson Company, Fort Washington, Pennsylvania, April 1980.
4. Boriah, S.; Chandola, V.; Kumar, V. Similarity Measures for Categorical Data: A Comparative Evaluation. In: SDM, pp. 243–254. SIAM, Philadelphia (2008).
5. Cunningham, Padraig. *A Taxonomy of Similarity Mechanisms for Case-Based Reasoning*; Technical Report UCD-CSI-20080-11; University College Dublin, January 6, 2008.
6. Denning, D. E. An Intrusion-detection Model. *IEEE Transactions on Software Engineering* **February 1987**, *13*(2).
7. Doyle, Donal. A Knowledge-Light Mechanism for Explanation in Case-Based Reasoning," University of Dublin, Trinity College. Department of Computer Science, Doctoral Thesis TCD-CS-2005-71, 2005.
8. Gafni, R. Framework for Quality Metrics in Mobile-Wireless. (K. Lynch, Ed.) *Interdisciplinary Journal of Information, Knowledge, and Management* **2008**, *3*, 23–38.
9. Gafni, R. Quality Metrics for PDA-based M-learning Information Systems. (A. Koohang, Ed.) *Interdisciplinary Journal of E-Learning and Learning Objects* **2009**, *5*, 359–378.
10. Junker, M.; Hoch, R.; Dengel, A. On the Evaluation of Document Analysis Components by Recall, Precision, and Accuracy. *Document Analysis and Recognition, 1999. ICDAR '99. Proceedings of the Fifth International Conference on*, pp.713–716, 20–22 Sep 1999.
11. Kolodner, Janet. Reconstructive Memory: A Computer Model. *Cognitive Science* 7 **1983**, *4*, 281–328.
12. Lawrie, Dawn; Feild, Henry; Binkley, David. Syntactic Identifier Conciseness and Consistency, In *Proceedings of the Sixth IEEE International Workshop on Source Code Analysis and Manipulation* (SCAM '06). IEEE Computer Society, Washington, DC, USA, pp. 139–148, 2006.

13. Lebowitz, Michael. 1983. Memory-based Parsing. *Artificial Intelligence* **November 1983**, *21* (4), 363–404.

14. Long, Jidong; Stoecklin, Sara; Schwartz, Daniel G.; Patel, Mahesh. Adaptive Similarity Metrics in Case-based Reasoning, *The 6th IASTED International Conference on Intelligent Systems and Control* (ISC 2004), August 23–25, 2004, Honolulu, Hawaii, pp. 260–265.

15. De Mantaras, Ramon Lopez; McSherry, David; Bridge, Derek; Leake, David; Smyth, Barry; Craw, Susan; Faltings, Boi; Maher, Mary Lou; Cox, Michael T.; Forbus, Kenneth; Leane, Mark; Aamodt, Agnar; Watson, Ian. Retrieval, Reuse, Revision and Retention in Case-based Reasoning. *Knowl. Eng. Rev.* **September 2005**, *20* (3), 215–240.

16. Osborne, H.; Bridge, D. Models of Similarity for Case-Based Reasoning. *Proc. Interdisciplinary Workshop Similarity and Categorisation,* pp. 173–179, 1997.

17. Ranganathan, A.; Ronen, R. Information-Theory Based Measure of Similarity Between Instances in Ontology, International Business Machines Corporation, United States Patent #7,792,838 B2, Sep. 7, 2010.

18. Roesch, Martin Roesch. Snort - Lightweight Intrusion Detection for Networks. In *Proceedings of the 13th USENIX conference on System administration* (LISA '99). USENIX Association, Berkeley, CA, USA, 229–238, 1999.

19. Schank, Roger. *Dynamic Memory: A Theory of Learning in Computers and People*; (New York: Cambridge University Press, 1982.

20. Silva, Luís A. L.; Buxton, Bernard F.; Campbell, John A. Enhanced Case-Based Reasoning through Use of Argumentation and Numerical Taxonomy. *The 20th International Florida Artificial Intelligence Research Society Conference* (FLAIRS-20), Special Track on Case-Based Reasoning. AAAI Press, Key West, Florida, 2007, 423–428.

21. Soh, L. K.; Blank, T. Integrating Case-Based Reasoning and Meta-Learning for a Self-Improving Intelligent Tutoring System. *Int. J. Artif. Intell. Ed.* **January 2008**, *18* (1) 27–58.

22. Stahl, A. Learning of Knowledge-Intensive Similarity Measures in Case-Based Reasoning. PHD-Thesis, dissertation.de, Technische Universität Kaiserslautern, 2004.

23. Paxson, Vern. Bro: A System for Detecting Network Intruders in Real-Time. *Computer Networks* **1999**, *31* (23–24), 2435–2463.

24. Sterling, W. M; Ericson, B. J. Case-Based Reasoning Similarity Metrics Implementation Using User Defined Functions," NCR Corp., United States Patent # 7,136,852 B1, Nov. 14, 2006.

25. Sun, Z.; Finnie, G.; Weber, K. Abductive Case Based Reasoning. *International Journal of Intelligent Systems* **2005**, *20* (9), 957–983.

26. van Rijsbergen, C. 1979. *Information Retrieval*. London: Butterworths.

27. Vesanto, J.; Hollmén, J.; Abraham, A.; Koppen, M. An Automated Report Generation Tool for the Data Understanding Phase. *Hybrid Information Systems* **2002**, 611–626.

28. Wang, Hui; Dubitzky, Werner. A Flexible and Robust Similarity Measure Based on Contextual Probability. In *Proceedings of the 19th international joint conference on Artificial intelligence* (IJCAI'05). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 27–32, 2005.

29. http://www2.dir.state.tx.us/SiteCollectionDocuments/Security/Incident%20Management/securityincident_reportingform.doc. (Accessed on 06/06/2011)

30. http://bestpractical.com/static/rtir/rtir-presentation.pdf. (Accessed on 06/06/2011)

31. http://www.ja.net/services/csirt/wp-content/uploads/rtir-incident-handling.pdf. (Accessed on 06/06/2011)

32. http://www.dfki.uni-kl.de/~aabecker/Mosbach/Bergmann-CBR-Survey.pdf (Accessed on 06/08/2011)

33. http://www.bleedingsnort.com/downloads/ (Accessed on 06/21/2011)

34. https://www.snort.org/snort-rules/#rules (Accessed on 06/21/2011)

Appendix A. Example Snort Rules

http://www.bleedingsnort.com/downloads/

```
alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg: "BLEEDING-EDGE P2P Ares
traffic"; flow: established; content:"User-Agent\: Ares";
reference:url,www.aresgalaxy.org; classtype: policy-violation; sid: 2001059;
rev:4; )

alert tcp $HOME_NET any -> $EXTERNAL_NET 6112 (msg:"BLEEDING-EDGE GAMES
Battle.net Warcraft 3\: The Frozen throne login"; flow:established,to_server;
content:"|FF 50|"; depth:2; content:"PX3W"; offset:12; depth:12; classtype:
policy-violation; sid:2002108; rev:2;)

alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS (msg:
"BLEEDING-EDGE MALWARE 180solutions Update Engine"; flow:
to_server,established; content:"GET"; depth: 3;
content:"Host|3a|"; within: 300;
content:"ping.180solutions.com"; within: 40;
reference:url,www.safer-
networking.org/index.php?page=threats&detail=212; classtype:
trojan-activity; sid: 2000930; rev:4; )

alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg: "BLEEDING-EDGE F5 BIG-IP
3DNS TCP Probe 1"; id: 1; dsize: 24; flags: S,12; content:"|00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|"; window: 2048;
classtype: misc-activity;
reference:url,www.f5.com/f5products/v9intro/index.html; sid: 2001609; rev:9;
)

alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg: "BLEEDING-EDGE EXPLOIT
Buffer Overflow Exploit in Adobe Acrobat Reader"; flow: established;
content:"URI/URI"; nocase; pcre:"/URI/URI\(mailto\:[^"]*"[^"]*"x[\d]{3}/i";
reference:url,www.securiteam.com/securitynews/5WP080AAKK.html;
classtype:shellcode-detect; sid: 2001049; rev:6; )

alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS (msg: "BLEEDING-EDGE DOS
HTTP GET with newline appended"; flowbits:noalert; flow:
to_server,established; content:"GET / HTTP/1.0|0a|"; offset: 0; depth: 15;
flowbits:set,http.get; reference:cve,2004-0942; classtype: attempted-dos;
sid: 2001635; rev:6; )

alert tcp $HOME_NET 21 -> $EXTERNAL_NET any (msg:"BLEEDING-EDGE ATTACK
RESPONSE Potential FTP Brute-Force attempt"; flow:from_server,established;
content:"530 "; pcre:"/^530s+(Login|User)/smi"; classtype:unsuccessful-user;
threshold: type threshold, track by_dst, count 5, seconds 120; sid:2002383;
rev:1;)

alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg: "BLEEDING-EDGE
WEB WebAPP Apage.CGI Remote Command Execution Attempt"; flow:
```

```
to_server,established; uricontent:"/apage.cgi?f="; nocase;
pcre:"/(\.\|.+\|)/"; reference:bugtraq,13637; classtype: web-application-
attack; sid: 2001945; rev:4; )
```

Appendix B. Snort Official Rule Documentation Example 1

http://www.bleedingsnort.com/downloads/

8543.txt 000644 002050 002050
3610 10530425761 16427 0 ustar
00nhoughton nhoughton 000000 000000 doc/signatures
Rule:

--

Sid:

8543

--

Summary:

This event is generated when activity relating to the spyware application “deluxecommunications” is detected.

--

Impact:

Unkown. Possible information disclosure, violation of privacy, possible violation of policy.

--

Detailed Information:

Spyware is malicious software running on a host that may intercept or take information from the host system without a users consent or knowledge. Spyware is also capable of using a hosts Internet connection without the 29oughton29 or consent of the user, in order to deliver that information to an unauthorized third party.

This software not only uses available bandwidth on a network connection but also consumes system resources to the point of making the host unusable in some cases.

Spyware can be classified into multiple categories depending on the behavior of the software.

In particular this event indicates that the software detected is trackware. Trackware programs are used to send information about the user to third parties without the users consent.

--

Affected Systems:

Windows operating system. All versions.

--

Attack Scenarios:

Spyware is often installed as a byproduct of the installation of another piece of software without the users knowledge. It may also be installed without the users consent via a vulnerability in software on the host machine, for example, by visiting a malicious website that utilizes a vulnerability in a browser application to install the spyware.

--

Ease of Attack:

Simple. This is spyware activity.

--

False Positives:

None known.

--

False Negatives:

None known.

--

Corrective Action:

Disallow the installation of software by unprivileged users.

Make use of anti-spyware products to remove the spyware from the affected system.

--

Contributors:

Sourcefire Vulnerability Research Team

Alex Kirk <alex.kirk@sourcefire.com>

Nigel Houghton <nigel.houghton@sourcefire.com>

--

Additional References

--

INTENTIONALLY LEFT BLANK.

Appendix C. Snort Official Rule Documentation Example 2

http://www.bleedingsnort.com/downloads/

8743.txt 000644 002050 002050
4504 10530425767 16442 0 ustar
00nhoughton nhoughton 000000 000000 doc/signatures
Rule:

--

Sid:

8743

--

Summary:

This event is generated when an attempt is made to return to a web client a file with a Class ID (CLSID) embedded in the file.

--

Impact:

A successful attack may result in the execution of code of the attackers choosing possibly leading to control of the target machine.

--

Detailed Information:

Internet Explorer does not correctly handle ActiveX controls. Certain COM objects can be called by Internet Explorer and executed as ActiveX controls. When this is achieved, it may be possible for an attacker to overwrite portions of memory and execute code of their choosing.

There are multiple CLSIDs associated with a COM component that could be used for malicious purposes. This event is generated when the CLSID for DirectAnimation.DAFontStyle.1 is detected in data being returned to a client system from a server.

These access rules alert on attempts to access certian CLSIDs that could potentially be used to exploit ActiveX based vulnerabilities. These CLSIDS fall into one of the following categories.

1. Microsoft has deprecated the CSLID and has suggested to all web developers that this CLSID no longer be used.

2. Microsoft has disabled the CLSID and it can no longer be used to actually execute an ActiveX object.

3. The CLSID is a known bad CLSID and has been removed from updated versions of the application that use this ActiveX object.

4. The ActiveX object is not safe for scripting and could be vulnerable to a security issue that may result in a buffer overflow or loss of system functionality.

This event indicates that the identifier for the component DirectAnimation.DAFontStyle.1 was detected.

--

Affected Systems:

Microsoft Internet Explorer 6 and prior

--

Attack Scenarios:

An attacker can entice a user to visit a web server that will return a malicious file with a file name that contains a CLSID, perhaps enabling the execution of the malicious code when the file is opened.

--

Ease of Attack:

Simple. Exploit code is publicly available.

--

False Positives:

None known.

--

False Negatives:

None known.

--

Corrective Action:

Upgrade to the latest non-affected version of the software.

Apply the appropriate vendor supplied patches.

--

Contributors:

Sourcefire Vulnerability Research Team

Alex Kirk <akirk@sourcefire.com>

Nigel Houghton <nigel.houghton@sourcefire.com>

--

Additional References

--

INTENTIONALLY LEFT BLANK.

Appendix D. Snort Official Rule Documentation Example 3

http://www.bleedingsnort.com/downloads/

16850.txt 000644 002050 002050
2722 11524301405 16502 0 ustar
00nhoughton nhoughton 000000 000000 doc/signatures
Rule:

--

Sid:

16850

--

Summary:

This event is generated when a system connects to a known-malicious domain.

--

Impact:

The system connecting to the domain is likely infected with malware, or may have been exposed to malicious code.

--

Detailed Information:

The Sourcefire VRT maintains a set of domain names automatically visited by malware-infected machines with no human interaction; all traffic from the machines is known to be generated by malware. After applying an extensive whitelist, the VRT pulls out the most commonly visited domains and adds them to its blacklist.rules category. The supplied reference lists the md5sum of the piece of malware used to infect the machine that generated the traffic in question.

--

Affected Systems:

All Windows versions

--

Attack Scenarios:

This alert is likely generated by malware communicating with the Internet in the background on an infected machine.

--

Ease of Attack:

Easy; the machine is likely already infected.

--

False Positives:

Occasionally the VRT's whitelisting process misses legitimate servers, particularly ad servers. If you feel that the domain here is legitimate, please notify us at fp@sourcefire.com so we can investigate.

--

False Negatives:

None Known

--

Corrective Action:

Scan the machine in question for malicious software. The VRT recommends ClamAV for Windows 3.0.

--

Contributors:

Sourcefire Vulnerability Research Team

--

Additional References:

http://labs.snort.org/docs/16850.html

--

INTENTIONALLY LEFT BLANK

Appendix E. Example Snort Rules

https://www.snort.org/snort-rules/#rules

alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS (msg:"BACKDOOR Backdoor.Win32.Qakbot.E - initial load"; flow:to_server,established; content:"/cgi-bin/jl/jloader.pl"; nocase; http_uri; reference:url,www.threatexpert.com/report.aspx?md5=8171d3223f89a495f98c4e3a65537b8f; classtype:trojan-activity; sid:16804; rev:2;)

alert tcp $HOME_NET any -> $EXTERNAL_NET 21 (msg:"BACKDOOR Backdoor.Win32.Qakbot.E - FTP upload seclog"; flow:to_server,established; content:"seclog"; fast_pattern:only; nocase; pcre:"/seclog_[a-z]{5}\d{4}_\d{10}\x2Ekcb/smi"; reference:url,www.threatexpert.com/report.aspx?md5=8171d3223f89a495f98c4e3a65537b8f; classtype:trojan-activity; sid:16806; rev:1;)

alert udp $EXTERNAL_NET 53 -> $HOME_NET any (msg:"DNS SPOOF query response PTR with TTL of 1 min. and no authority"; flow:to_client; content:"|85 80 00 01 00 01 00 00 00 00|"; content:"|C0 0C 00 0C 00 01 00 00 00|<|00 0F|"; fast_pattern:only; metadata:policy security-ips drop; classtype:bad-unknown; sid:253; rev:9;)

alert udp $EXTERNAL_NET 53 -> $HOME_NET any (msg:"DNS SPOOF query response with TTL of 1 min. and no authority"; flow:to_client; content:"|81 80 00 01 00 01 00 00 00 00|"; content:"|C0 0C 00 01 00 01 00 00 00|<|00 04|"; fast_pattern:only; metadata:policy security-ips drop; classtype:bad-unknown; sid:254; rev:9;)

alert ip $EXTERNAL_NET any -> $HOME_NET any (msg:"SHELLCODE x86 fldz get eip shellcode"; content:"|D9 EE D9|t|24 F4|X"; metadata:policy balanced-ips drop, policy security-ips drop; classtype:shellcode-detect; sid:14986; rev:4;)

alert tcp $EXTERNAL_NET any -> $HOME_NET [139,445] (msg:"SHELLCODE x86 win2k-2k3 decoder base shellcode"; flow:to_server,established; content:"|C7 0B|GGGG|81|7";

content:"u|F4|"; within:2; distance:4; metadata:policy balanced-ips drop, policy security-ips drop; reference:bugtraq,19409; reference:cve,2006-3439; classtype:attempted-user; sid:15902; rev:1;)

alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS (msg:"WEB-CGI Majordomo2 http directory traversal attempt"; flow:established,to_server; content:"mj_wwwusr"; fast_pattern; nocase; http_uri; content:"extra="; distance:0; nocase; http_uri; content:"../../.."; distance:0; http_uri; metadata:policy balanced-ips drop, policy security-ips drop, service http; reference:bugtraq,46127; reference:cve,2011-0049; classtype:web-application-attack; sid:18761; rev:1;)

alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS (msg:"WEB-CGI Symantec IM Manager LoggedInUsers.lgx definition file multiple SQL injections attempt"; flow:to_server,established; content:"/IMManager/rdPage.aspx?"; fast_pattern:only; http_uri; pcre:"/\x2FIMManager\x2FrdPage\x2Easpx\x3F.*?(loginTimeStamp|dbo|dateDiffParam|whereClause)\x3D[^\x26]*?(\x3B|\x23|\x2D{2})/sU"; metadata:policy security-ips

alert tcp $EXTERNAL_NET any -> $HOME_NET 5000 (msg:"SCAN UPnP service discover attempt"; flow:to_server,established; content:"M-SEARCH "; depth:9; content:"ssdp|3A|discover"; fast_pattern:only; classtype:network-scan; sid:8081; rev:2;)

alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS (msg:"SCAN Proxyfire.net anonymous proxy scan"; flow:to_server,established; content:"proxyfire.net/fastenv"; nocase; http_uri; reference:url,www.proxyfire.net/index.php; classtype:network-scan; sid:18179; rev:2;)

Appendix F. Snort Official Rule Message Example

https://www.snort.org/snort-rules/#rules

215 || BACKDOOR MISC Linux rootkit attempt
216 || BACKDOOR MISC Linux rootkit satori attempt || arachnids,516
221 || DDOS TFN Probe || arachnids,443 || cve,2000-0138
222 || DDOS tfn2k icmp possible communication || arachnids,425 || cve,2000-0138
253 || DNS SPOOF query response PTR with TTL of 1 min. and no authority
262 || DNS EXPLOIT x86 Linux overflow attempt
264 || DNS EXPLOIT x86 Linux overflow attempt
850 || WEB-CGI wais.pl access
15883 || EXPLOIT SAPLPD 0x01 command buffer overflow attempt || bugtraq,27613 || cve,2008-0621
16809 || BOTNET-CNC known command and control channel traffic || url,labs.snort.org/docs/16809.html

INTENTIONALLY LEFT BLANK.

Appendix G. Example of Fired Snort Alerts

http://hintsforums.macworld.com/archive/index.php/t-36315.html

[**] [1:2123:2] ATTACK-RESPONSES Microsoft cmd.exe banner [**]
[Classification: Successful Administrator Privilege Gain] [Priority: 1]
03/09-19:43:56.034979 66.59.111.182:80 -> xxx:60134
TCP TTL:45 TOS:0x0 ID:45583 IpLen:20 DgmLen:1492 DF
A* Seq: 0x5314DE4 Ack: 0xC70EBBC2 Win: 0x198C TcpLen: 32
TCP Options (3) => NOP NOP TS: 193196204 1313605945
[Xref => http://cgi.nessus.org/plugins/dump.php3?id=11633]

[**] [1:1200:10] ATTACK-RESPONSES Invalid URL [**]
[Classification: Attempted Information Leak] [Priority: 2]
03/09-19:44:13.338632 64.94.117.8:80 -> xxx:60135
TCP TTL:41 TOS:0x0 ID:30264 IpLen:20 DgmLen:1492 DF
A* Seq: 0x29121B52 Ack: 0x839B334C Win: 0x1920 TcpLen: 20
[Xref => http://www.microsoft.com/technet/security/bulletin/MS00-063.mspx]

[**] [1:1201:7] ATTACK-RESPONSES 403 Forbidden [**]
[Classification: Attempted Information Leak] [Priority: 2]
03/09-19:52:42.185750 199.107.65.177:80 -> xxx:60197
TCP TTL:40 TOS:0x0 ID:64774 IpLen:20 DgmLen:1420 DF
A* Seq: 0xB38BE67B Ack: 0x1EFCDBD7 Win: 0x2180 TcpLen: 32
TCP Options (3) => NOP NOP TS: 1004293146 1313606996

[**] [1:497:11] ATTACK-RESPONSES file copied ok [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
03/09-20:34:00.379435 205.206.231.13:80 -> xxx:61607
TCP TTL:43 TOS:0x0 ID:54613 IpLen:20 DgmLen:1492 DF
A* Seq: 0xD8357B2D Ack: 0x49F73C5E Win: 0x2220 TcpLen: 20
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=2000-0884][Xref =>
http://www.securityfocus.com/bid/1806]

[**] [1:498:6] ATTACK-RESPONSES id check returned root [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
03/09-20:46:19.176514 64.151.140.130:80 -> xxx:62038
TCP TTL:52 TOS:0x0 ID:42702 IpLen:20 DgmLen:1492 DF
A* Seq: 0x13E85710 Ack: 0x6F91FBB5 Win: 0x1920 TcpLen: 32
TCP Options (3) => NOP NOP TS: 1894901317 1313613431

[**] [1:497:11] ATTACK-RESPONSES file copied ok [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
03/09-20:52:51.466517 205.206.231.15:80 -> xxx:62057

TCP TTL:43 TOS:0x0 ID:48885 IpLen:20 DgmLen:1492 DF
A* Seq: 0x37048CE9 Ack: 0x57D967E Win: 0x1A5E TcpLen: 20
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=2000-0884][Xref =>
http://www.securityfocus.com/bid/1806]

Appendix H. Example Snort-Rule Messages

http://www.bleedingsnort.com/downloads/

BLEEDING-EDGE MALWARE 180solutions Update Engine
BLEEDING-EDGE MALWARE 180solutions Spyware (tracked event reported)
BLEEDING-EDGE MALWARE 180solutions Spyware (action url reported)
BLEEDING-EDGE MALWARE 180solutions Spyware Reporting
BLEEDING-EDGE MALWARE 180solutions Spyware Keywords Download
BLEEDING-EDGE MALWARE 180solutions Spyware Install
BLEEDING-EDGE MALWARE 180solutions Spyware Defs Download
BLEEDING-EDGE MALWARE 180solutions Spyware config Download
BLEEDING-EDGE MALWARE 180solutions Spyware versionconfig POST
BLEEDING-EDGE MALWARE Spyware 2020
BLEEDING-EDGE MALWARE 2020search Update Engine
BLEEDING-EDGE MALWARE Advertising.com Agent
BLEEDING-EDGE MALWARE Advertising.com Data Post (villains)
BLEEDING-EDGE MALWARE Advertising.com Data Post (cakedeal)

NO. OF COPIES	ORGANIZATION
1 PDF	ADMNSTR DEFNS TECHL INFO CTR ATTN DTIC OCP
1	US ARMY CYBER COMMAND ATTN 24 C PRESSLEY
1	US ARMY RSRCH LAB ATTN RDRL CIN D L M MARVEL
1	US ARMY RSRCH LAB ATTN RDRL CIN D T PARKER
1	US ARMY RSRCH LAB ATTN RDRL CIN S C SMITH
4	US ARMY RSRCH LAB ATTN RDRL CIN D B RESCHLY ATTN RDRL CIN D C ELLIS ATTN RDRL CIN D J COLE ATTN RDRL CIN D D KELLY
1	US ARMY RSRCH LAB ATTN RDRL SLE I A REVILLA
1	US ARMY RSRCH LAB ATTN RDRL SLE I D LANDIN
11	US ARMY RSRCH LAB ATTN RDRL CIN A KOTT ATTN RDRL CIN D H CAM ATTN RDRL CIN D M SHEVENELL ATTN RDRL CIN D P GUARINO ATTN RDRL CIN D R HARANG ATTN RDRL CIN D R PINO ATTN RDRL CIN D S HUTCHINSON ATTN RDRL CIN S C ARNOLD ATTN RDRL CIN S R ERBACHER ATTN IMAL HRA MAIL & RECORDS MGMT ATTN RDRL CIO LL TECHL LIB